Kragujevac J. Math. 27 (2005) 91–100.

PARALLEL ALGORITHM FOR SOLVING THE BLACK-SCHOLES EQUATION

Ioana Chiorean

Babeş-Bolyai University, Faculty of Mathematics
Kogălniceanu 1, 3400 Cluj-Napoca, Romania
(e-mail: ioana@cs.ubbcluj.ro)

(Received March 15, 2005)

Abstract. The aim of this paper is to study the possibility of obtaining the numerical solution of the Black-Scholes equation in parallel, by means of several processors, using the finite difference method. A comparison between the complexity of the parallel algorithm and the serial one is given.

1. INTRODUCTION

It is well-known that the Black-Scholes formula is used in computing the value of an option. In some cases, e.g. a European options, it gives exact solutions, but for others, more complex, numerical attempts are made in order to obtain an approximation of the solutions. Several numerical methods are used for solving the Black-Scholes equation. So, in [7], the finite element method is used, starting from the difusion equation. In [5], a novel afaptive radial basis function scheme based on the radial basis function methods is presented.

In [6], discrete symmetries of partial differential equations are used to create efficient numerical methods. They are associated also for the Black-Scholes equation.

In [4], a method for solving the generalized Black-Scholes formula, based on so-called additive operator splitting technique, is given.

2. THE FULL-IMPLICIT SCHEME FOR THE BLACK-SCHOLES FORMULA

As in [7], the Black-Scholes equation and the boundary condition for a European call with value $C(S,t)$ are:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{1}$$

with

$$C(0,t) = 0, \quad C(S,t) \sim S \text{ as } S \to \infty \tag{2}$$

and

$$C(S,T) = \max\{S - E, 0\} \tag{3}$$

where S is the current value of the underlying asset, t is the time, σ is the volatility of the underlying asset, E, the exercise price, r is the interest rate and T, the expiry.

In order to obtain a numerical approximation of the solution for (1), with the boundary conditions (2) and (3), by means of the finite difference method, a mesh of points is used, as in Figure 1.

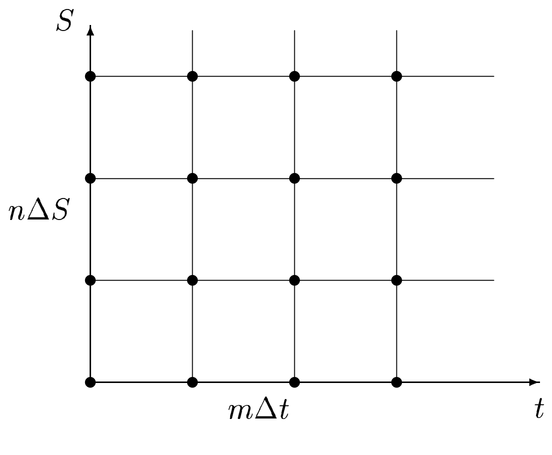

Figure 1. The mesh for a finite difference approximation.

The S-axis is divided into equally spaced nodes a distance ΔS apart, and the t-axis into equally spaced nodes a distance Δt apart. This divides the (S,t) plane into a mesh, where the mesh points have the form $(n\Delta S, m\Delta t)$.

In what follows, we denote

$$C_n^m = C(n\Delta S, m\Delta t)$$

for the value of $C(S,t)$ at the mesh point $(n\Delta S, m\Delta t)$.

Further, using a backward difference approximation for $\frac{\partial C}{\partial t}$, the symmetric centered difference approximation for $\frac{\partial^2 C}{\partial S^2}$ and the forward difference approximation for $\frac{\partial C}{\partial S}$, we get the following discretized form for the equation (1):

$$\frac{C_n^m - C_n^{m-1}}{\Delta t} + \frac{1}{2}\sigma^2 S^2 \frac{C_{n+1}^m - 2C_n^m + C_{n-1}^m}{(\Delta S)^2} + rS\frac{C_{n+1}^m - C_n^m}{\Delta S} - rC_n^m = 0 \qquad (4)$$

Making some calculus, we get:

$$C_n^{m-1} = \alpha C_{n+1}^m + \beta C_n^m + \gamma C_{n-1}^m \qquad (5)$$

where

$$\alpha = \frac{rS\Delta t\Delta S + \frac{1}{2}\sigma^2 S^2 \Delta t}{(\Delta S)^2}$$

$$\beta = \frac{(\Delta S)^2 - \sigma^2 S^2 \Delta t - rS\Delta t\Delta S - r\Delta t(\Delta S)^2}{(\Delta S)^2}$$

$$\gamma = \frac{1}{2}\sigma^2 S^2 \frac{\Delta t}{(\Delta S)^2}$$

For the discretized boundary conditions (2) and (3), as in [7], we assume that we can truncate the infinite mesh at $S = N^-\Delta S$ and $S = N^+\Delta S$ and take N^- and N^+ sufficiently large so that no significant errors are introduced. Then,

$$\begin{aligned} C_{N^-}^m &= C_{-\infty}(N^-\Delta S, m\Delta t), \quad 0 < m \le M \\ C_{N^+}^m &= C_{\infty}(N^+\Delta S, m\Delta t), \quad 0 < m \le M \end{aligned} \qquad (6)$$

and

$$C_n^0 = C_0(n\Delta S), \quad N^- \le n \le N^+ \qquad (7)$$

The problem is, then, to find C_n^m for $m \geq 1$ and $N^- < n < N^+$, from (5).

It is clear, as for every implicite scheme, that C_n^m (at the moment m) depends on three values from the moment $(m+1)$, as in Figure 2.

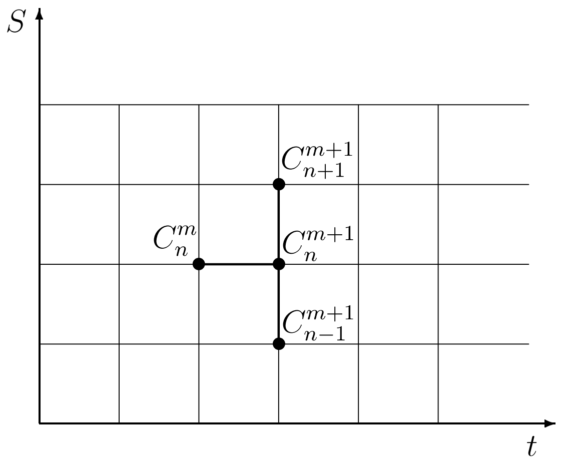

Figure 2. Implicit finite-difference discretization.

We can write (5) as the linear system:

$$\begin{bmatrix} \beta & \alpha & 0 & \dots & 0 \\ \gamma & \beta & \alpha & \dots & 0 \\ 0 & \gamma & \beta & \dots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ 0 & 0 & \dots & \gamma & \beta \end{bmatrix} * \begin{bmatrix} C_{N^-+1}^m \\ \vdots \\ C_0^m \\ \vdots \\ C_{N^+-1}^m \end{bmatrix} = \begin{bmatrix} C_{N^-+1}^{m-1} \\ \vdots \\ C_0^{m-1} \\ \vdots \\ C_{N^+-1}^{m-1} \end{bmatrix} \tag{8}$$

Or, in matriceal form:

$$A \cdot C^m = C^{m-1}, \tag{9}$$

where C^m and C^{m-1} denote the $(N^+ - N^- - 1)$-dimensional vectors

$$C^m = (C_{N^-+1}^m, \dots, C_{N^+-1}^m), \quad C^{m-1} = (C_{N^-+1}^{m-1}, \dots, C_{N^+-1}^{m-1})$$

and A is the $(N^+ - N^- - 1)$-square matrix given in (8), which, according with [7] is invertible and so

$$C^m = A^{-1} \cdot C^{m-1} \tag{10}$$

where A^{-1} is the inverse of A.

We can therefore find C^m by given C^{m-1}, which in turn may be found from C^{m-2} and the boundary conditions. As the initial conditions gives C^0, we can find each C^m step by step, or, according with the relation:

$$C^m = (A^{-1})^m \cdot C^0. \tag{11}$$

3. PRACTICAL CONSIDERATIONS

In practice, there are far more efficient solution techniques than matrix inversion, due to the property of A being tridiagonal. Then, methods like LU decomposition or SOR are applied directly to (10), and the execution time is $O(N)$ per solution. In order to compute A^{-1}, one needs $O(N^2)$ operation and others $O(M^2)$ to find $(A^{-1})^m$, using one processor, so in a serial manner. But with several processors under a convenient network, we show in what follows that we can obtain a time of execution $O(N)$, to compute the inverse A^{-1}.

4. PARALLEL ALGORITHM FOR COMPUTING THE NUMERICAL SOLUTION

4.1. THE METHOD OF ELEMENTARY TRANSFORMATION FOR THE INVERSE MATRIX

We use the method of elementary transformation to compute the inverse matrix, A^{-1} (see [2], [3]). In few words, we start from the matrix A', which is obtained from A and a unit matrix, written on the right side of A, as follows:

$$A' = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2N} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{N1} & a_{N2} & \dots & a_{NN} & 0 & 0 & \dots & 1 \end{bmatrix}$$

Note. For the sake of the clearness, we denote by a_{ij}, $i,j=\overline{1,N}$ all the elements of matrix A, it means α,β,γ and 0.

Further, making elementary transformations only on the lines of A', after several steps, we bring it to the form A'', where

$$A''=\begin{bmatrix} 1 & 0 & \dots & 0 & b_{11} & b_{12} & \dots & b_{1N} \\ 0 & 1 & \dots & 0 & b_{21} & b_{22} & \dots & b_{2N} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & b_{N1} & b_{N2} & \dots & b_{NN} \end{bmatrix}$$

The part $\begin{bmatrix} b_{11} & b_{12} & \dots & b_{1N} \\ b_{21} & b_{22} & \dots & b_{2N} \\ \dots & \dots & \dots & \dots \\ b_{N1} & b_{N2} & \dots & b_{NN} \end{bmatrix}$ represents A^{-1}.

The computation is made in the following manner:

Step 1.

$$A'=\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} & 1 & 0 & \dots & 0 \\ a_{21} & a_{22} & \dots & a_{2n} & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} & 0 & 0 & \dots & 1 \end{bmatrix}\sim$$

$$\sim\begin{bmatrix} 1 & a'_{12} & \dots & a'_{1n} & b'_{11} & b'_{12} & \dots & b'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} & b'_{21} & b'_{22} & \dots & b'_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a'_{n2} & \dots & a'_{nn} & b'_{n1} & b'_{n2} & \dots & b'_{nn} \end{bmatrix}$$

where

$$a'_{1j}=a_{1j}/a_{11},\ j=\overline{1,n},\quad b'_{1j}=b_{1j}/a_{11},\ j=\overline{1,n}$$

(**Note.** We denote with b_{ij} the element of unit matrix.)

$$a'_{ij}=a_{ij}-a'_{1j}a_{i1},\quad b'_{ij}=b_{ij}-b'_{1j}a_{i1},\quad i=\overline{2,n},\ j=\overline{1,n}$$

Step 2.

$$\begin{bmatrix} 1 & a'_{12} & \dots & a'_{1n} & b'_{11} & b'_{12} & \dots & b'_{1n} \\ 0 & a'_{22} & \dots & a'_{2n} & b'_{21} & b'_{22} & \dots & b'_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & a'_{n2} & \dots & a'_{nn} & b'_{n1} & b'_{n2} & \dots & b'_{nn} \end{bmatrix}\sim$$

$$\sim\begin{bmatrix} 1 & 0 & a''_{13} & \dots & a''_{1n} & b''_{11} & b''_{12} & \dots & b''_{1n} \\ 0 & 1 & a''_{23} & \dots & a''_{2n} & b''_{21} & b''_{22} & \dots & b''_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & a''_{n3} & \dots & a''_{nn} & b''_{n1} & b''_{n2} & \dots & b''_{nn} \end{bmatrix}$$

where

$$a''_{2j} = a'_{2j}/a'_{22},\ j=\overline{1,n},\quad b''_{2j} = b'_{2j}/a'_{22},\ j=\overline{1,n}$$

$$a''_{ij} = a'_{ij} - a''_{2j}a'_{i2},\quad b''_{ij} = b'_{ij} - b''_{2j}a'_{i2},\quad i,j=\overline{1,n},\ i\neq 2$$

and so on, till the matrix has the final form

$$\begin{bmatrix} 1 & 0 & \dots & 0 & b^n_{11} & b^n_{12} & \dots & b^n_{1n} \\ 0 & 1 & \dots & 0 & b^n_{21} & b^n_{22} & \dots & b^n_{2n} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & b^n_{n1} & b^n_{n2} & \dots & b^n_{nn} \end{bmatrix}$$

and A^{-1} is read from the second part of this matrix:

$$A^{-1} = \begin{bmatrix} b^n_{11} & b^n_{12} & \dots & b^n_{1n} \\ \dots & \dots & \dots & \dots \\ b^n_{n1} & b^n_{n2} & \dots & b^n_{nn} \end{bmatrix}$$

4.2. THE PARALLEL METHOD

It is clear that, using only one processor to make all the computations, the time of execution is $O(N^3)$, because we have n steps and every step needs $O(N^2)$ operations to be computed. In order to reduce the execution time, we can use the parallel calculus.

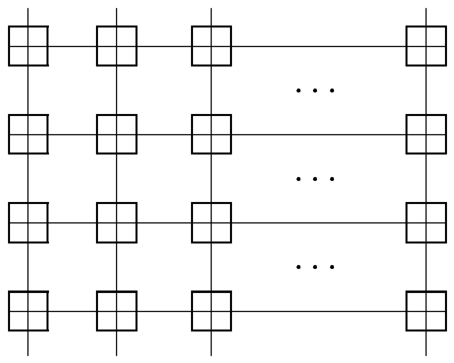

Figure 3. The lattice network.

Having in mind the previous method, we come back to the solving of system (1), using more than one processor. This can be done with $N \times 2N$ processors connected under a lattice network, like in Figure 3. In every node of the network there is a processor. According with [1], under this connectivity, every processor P_{ij} is connected and may transfer information with its four neighbourhood $P_{i-1,j}$, $P_{i+1,j}$, $P_{i,j-1}$, $P_{i,j+1}$, $i,j = \overline{1, N-1}$.

The computation of the inverse matrix A^{-1} can be made in the following manner:

Step 0. (Initialization)
$P_{ij} \leftarrow A', \quad i = \overline{1,N},\ j = \overline{1,2N}$ (every processor memorize the matrix A')
Step 1. In parallel do:
$P_{1j} \leftarrow a'_{1j} = a_{1j}/a_{11},\ j = \overline{1,N}$
$P_{1j} \leftarrow b'_{1j} = b_{1j}/a_{11},\ j = \overline{N+1,2N}$
$P_{ij} \leftarrow a'_{ij} = a_{ij} - a'_{1j}a_{i1},\ i = \overline{2,N},\ j = \overline{1,N}$
$P_{ij} \leftarrow b'_{ij} = b_{ij} - b'_{1j}a_{i1},\ i = \overline{2,N},\ j = \overline{N+1,2N}$
Step 2. In parallel do:
$P_{2j} \leftarrow a''_{2j} = a'_{2j}/a'_{22},\ j = \overline{1,N}$
$P_{2j} \leftarrow b''_{2j} = b'_{2j}/a'_{22},\ j = \overline{N+1,2N}$
$P_{ij} \leftarrow a''_{ij} = a'_{ij} - a''_{2j}a'_{i2},\ i,j = \overline{1,N},\ i \neq 2$
$P_{ij} \leftarrow b''_{ij} = b'_{ij} - b''_{2j}a'_{i2},\ i = \overline{1,N},\ j = \overline{N+1,2N},\ i \neq 2$

and so on, till step N, when the matrix in final form is obtained and the inverse matrix A^{-1} can be read.

The effort of computation is of order $O(N)$, because we still have N steps, but in parallel, every step takes the time for doing a division, a multiplication and a subtraction.

Note. Due to the fact that at step i, the line of processor P_{ij}, $j = \overline{1,2N}$ executes a division and all the other processors executes a subtraction and a multiplication, the problem of their syncronization has to be taken into account.

4.3. SOLVING THE FINAL SYSTEM IN PARALLEL

In the previous paragraph we show how the inverse matrix A^{-1} can be computed in parallel, with an execution time of order $O(N)$. In order to solve the system (11), which gives the final numerical solution for the Black-Scholes equation, we have to compute the power m of matrix A^{-1}. According with [1], this can be done in a logarithmic time, $O(\log_2 N)$ using a binary-tree connectivity among processors, like in Figure 4.

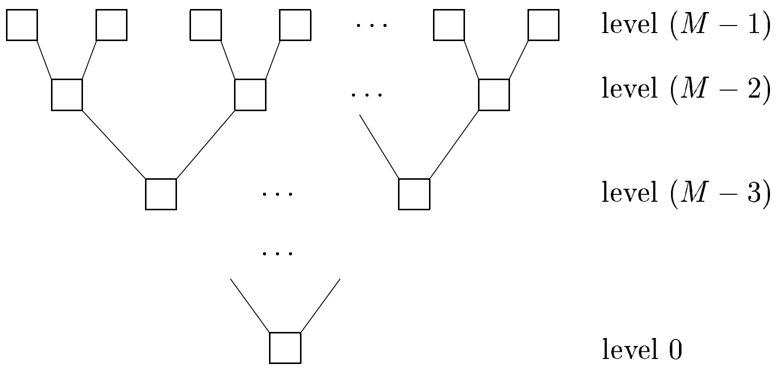

Figure 4. The binary-tree network.

Note. In every node of this network there is a processor. The idea of computation is the following:

Step 0. (Initialization)

Every processor leaf (at level $(M-1)$) memorizes the matrix A^{-1}.

Step 1. Every processor at level $(M-2)$ computes $(A^{-1})^2 = A^{-1} \cdot A^{-1}$.

Step 3. Every processor at level $(M-3)$ computes $(A^{-1})^4 = (A^{-1})^2 \cdot (A^{-1})^2$ and so on.

After $\log_2 M$ steps, the final result $(A^{-1})^M$ will be computed by the processor root.

5. CONCLUSIONS

We presented an algorithm which generates the numerical solution of the Black-Scholes equation for a European option in an execution time of order $O(N \cdot \log_2 M)$, using parallel calculus. The binary-tree network can be included in the lattice network, in order to use the same processors.

References

[1] I. Chiorean, *Calcul paralel*, Ed. Microinformatica, Cluj, 1994.

[2] Gh. Coman, *Analiză numerică*, Ed. Libris, 1992.

[3] Gh. Coman, D. Johnson, *Complexitatea algoritmilor*, Lito Univ. Babeş-Bolyai, Cluj, 1987.

[4] S. Kilianova, D. Ševčovič, *Analytical and Numerical Methods for Stock Index Derivative Pricing*, J. of Electrical Engineering, vol. **55**, Nr. 12/1, 2004, 1-5.

[5] M. B. Voc, I. Boztosun, D. Boztosun, *On the Numerical Solution of Black-Scholes Equation*, Proc. of Int. Workshop on Mesh Free Methods, 2003.

[6] Gh. Silberberg, *Discrete Symmetries of the Black-Scholes Equation*, site internet http://www.ceu.hu

[7] P. Wilmott, et al., *The Mathematics of Financial Derivatives*, Cambridge Univ. Press, 1995.